

An Empirical Comparison of Pre-Trained Models of Source Code

Changan Niu, Chuanyi Li, Vincent Ng, Dongxiao Chen, Jidong Ge, Bin Luo

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Human Language Technology Research Institute, University of Texas at Dallas, Richardson, Texas, USA

発表者: 稲葉 真幸

事前学習済みモデル

- 深層学習と学習データ

近年、深層学習がAI分野の自然言語処理やコンピュータビジョンなど様々に利用されている

× 欠点：タグが付いた学習データを大量に集める必要がある

- 事前学習済みモデル（PTM：Pre-Trained Models）

大量のデータから一般的なパターンを既に学習させてあるモデルのこと

ゼロからデータ収集する必要がなく、

利用する際に特化させたい特定のタスクに対してファインチューニングのみ行う

例）“自然言語”の事前学習モデル（PTM-NL）

BERT, GPT など

事前学習済みモデル

- “事前学習済みモデル” のソフトウェア工学への応用

初期は自然言語の事前学習モデルをソースコードについて再学習させていた

△ とはいえ、このままSEタスクに採用するのは理想的でない

- ソースコードの事前学習済みモデル（CodePTM）

事前学習モデルのうち、特にソースコードの特性

に基づき事前学習が行われているもの

ソフトウェア工学の研究者によって数多く開発された

同分野の多くのタスクをこなすことができる有用なツールとなった

固有の構文
意味構造など



“ソースコードの事前学習済みモデル”に対する理解

- ソースコードの事前学習モデル（CodePTM）

ソフトウェア工学の研究者によって数多く開発された

×これまでの研究では、これらモデルの比較が十分に行われておらず理解が限定的
事前学習済みモデルはさまざまなソフトウェア工学のタスクに適用可能だが
少数のタスクでしか評価、比較が行われていない

- 最近開発された19のソースコードの事前学習モデルを
13のソフトウェア工学のタスクによって体系的に比較する.
- CodePTM内でどのカテゴリに分類されているか、と
タスクの性能との間に相関関係があるかを調査.

CodePTMの分類

19のモデルをこれらの観点からカテゴライズ

- Arch. (アーキテクチャ)
LSTM : Long Short-Term Memory
TF : Transformer
TE : Transformer Encoder (エンコーダのみのトランスフォーマ)
TD : Transformer Decoder (デコーダのみのトランスフォーマ)
- Modality (モーダリティ) : PTMのデータ入力形式
NL : Natural Language
Code
複数形式
- Tasks (事前学習済みタスク)
☆後述
- PL (プログラミング言語)
一つの言語に対応
複数の言語に対応

Models	Arch.	Modality	Tasks	PL
RoBERTa [7]	TE	NL Text	MLM	-
GPT-2 [9]	TD	NL Text	ULM	-
BART [11]	TF	NL Text	DAE	-
T5 [10]	TF	NL Text	Seq2Seq MLM	-
SCELMo [52]	LSTM	Code	BiLM	Mono
CuBERT [12]	TE	Code	MLM + NSP	Mono
GPT-C [14]	TD	Code	ULM	Multi
C-BERT [13]	TE	Code	MLM	Mono
JavaBERT [15]	TE	Code	MLM	Mono
CodeGPT-adapted [35]	TD	Code	ULM	Multi
DeepDebug [16]	TF	Code	Seq2Seq MLM	Mono
CodeBERT [53]	TE	Code + Doc	MLM & RTD	Multi
GraphCodeBERT [54]	TE	Code + Doc + DFG Nodes	MLM + EP + NA	Multi
CugLM [40]	TE	Code	IMLM + NSP + ULM	Multi
DOBF [55]	TF	Code	MLM & Seq2Seq IMLM	Multi
T5-learning [56]	TF	Code & Doc	Seq2Seq MLM	Mono
PLBART [57]	TF	Code & Post	DAE	Multi
ProphetNet-Code [58]	TF	Code & Doc	FNP	Multi
CoText [59]	TF	Code + Doc	Seq2Seq MLM	Multi
TreeBERT [28]	TF	Code + AST Paths	TMLM + NOP	Multi
OSCAR [60]	TE	IR + AEI	MLM + CCL	Mono
CodeDisen [61]	LSTM	Code + AST Seq	VGVAE + CLR + PD + ACP	Multi
CodeT5 [62]	TF	Code + Doc	Seq2Seq MLM / IT / Seq2Seq IMLM / BDG	Multi
SynCoBERT [63]	TE	Code + Doc + AST Seq	MLM + IT + TEP + MCL	Multi
SPT-Code [29]	TF	Code + Names + AST Seq	CAP & MASS & MNG	Multi
UniXcoder [64]	TE	AST Seq + Doc	MLM / ULM / Seq2Seq MLM / MCL / CMG	Multi

事前学習済みタスク

モデルの事前学習時に使用されたタスク。評価に使うタスクとは異なる

- Type

C/NLA : Code / Natural Language Aware

CA : Code Aware Only

SA : Code Structure Aware Only

CMA : Cross Modal Aware Only

CN : Code NL

CS : Code Structure

CNS : Code-NL-Structure

- O. (Output)

G : Generative

C : Categorical

Type	O.	Task	Full Name and Description
C/NLA	G.	ULM [14]	Unidirectional LM: conditional on words that have already appeared, maximizes the conditional probability of all next words.
		FNP [58]	Future N-gram Prediction: conditional on words that have appeared, maximizes the conditional probability of all next N ($N > 1$) words.
		BiLM [52]	Bidirectional LM: apply ULM to the input and its reversion to maximize the conditional probability of each word in both directions.
		MLM [15]	Masked Language Model: predicts a certain percentage of tokens that have been randomly masked in the input. (Basic version of MLM)
		WWM [13]	Whole Word Masking: a variant of basic MLM, if parts of a word is masked, ensure all subwords/tokens in it be masked.
		MASS [29]	MAsked Seq2Seq: predicts 50% of the content that is randomly masked consecutively in the sentence in the encoder-decoder architecture.
		SMLM [56]	Seq2Seq MLM: sequentially predicts a set of token spans randomly masked in the input in the encoder-decoder framework.
	C.	DAE [57]	Denosing Auto-Encoding: recovers the original input from the one tampered by masking, deleting, and replacing tokens, etc.
		NSP [12]	Next Sentence Prediction: determines whether the two given sentences or logical lines of code appear consecutively in real world.
		RTD [53]	Replaced Token Detection: identifies whether a token in the input is a fake one that is produced by a small generator network.
CA	G.	IMLM [40]	Identifier MLM: predicts a certain percentage of identifiers that randomly masked in the code text (an adaption of basic MLM to code).
		SIMLM [62]	Seq2Seq IMLM: an adaptation of Seq2Seq MLM to source code that masks only a certain percentage of the identifiers in the code text.
	C.	IT/IP [62]	Identifier Tagging/Predicting: determines whether the input token at each position is an identifier or not via binary classification.
SA	C.	CCL [60]	Code Contrastive Learning: minimizes/maximizes the distances between the representations of similar/dissimilar code snippets.
		EP/TEP [54]	Edge Prediction: predicts the edges that are masked by randomly selecting source and target nodes in a DFG or AST.
		NOP [28]	Node Order Prediction: determines if a change occurs in an AST where the order of some randomly selected nodes are changed.
CMA	CN	BDG/CMG [62]	Bimodal Dual Generation/Cross Modal Generation: generates a Natural Language/Code if Code/Natural Language is given.
		MNG [29]	Method Name Generation: produces a name for the given method body by generating sub-tokens with the decoder sequentially.
	CS	CLR [61]	Cross-Language Reconstruction: generates a code snippet in one PL functionally equivalent to the given one in other PLs.
		TMLM [28]	Tree MLM: generates complete code from inputs where some terminal nodes/identifiers in ASTs/code are masked in encoder/decoder.
		VGVAE [61]	vMF-Gaussian Variational Autoencoder: disentangles code semantics from code syntax under the supervision of a masked AST.
		CAP [29]	Code-AST Prediction: determines whether the given code and AST in the input correspond to each other via binary classification.
		NA [54]	Node Alignment: predicts the masked edges connecting randomly sampled nodes in a DFG and its corresponding code token.
	C.	PD [61]	Posterior Distribution: minimizes the difference in semantics distributions of functionally equivalent code snippets in different PLs.
		ACP [61]	Attentive Code Position: predicts the node type in AST of a code token in the input through an attention mechanism.
		MCL [63]	Multi-modal Contrastive Learning: an adaptation of CCL to (Code,NL)/(Code,Structure) pairs where samples are no longer code pairs.
	CNS	C.	

評価に使用するタスク・データセット

ソフトウェア工学に関連する13の評価用タスクを用意

タスクは以下によって分類

- Und. : Understanding, 理解
- Gen. : Generation, 生成
- C : Code, コード
- NL : Natural Language, 自然言語
- V : Extracted/Predicted Value 抽出/予測値

メトリクス

- Acc : Accuracy
- F1
- P : Precision
- R : Recall
- MRR : Mean Reciprocal Rank
- MAP : Mean Average Precision

Type	I-O	Task	Ab.	ID: Dataset	Metrics
Und.	C-V	Defect Detection	DD	D1: Devign [30]	Acc
				D2: DeepBugs [31]	Acc
				V1: VMC [12]	Acc
				W1: WBO [12]	Acc
				S3: SO [12]	Acc
	C-V	Clone Detection	CD	B1: BigCloneBench [32]	F1
				C2: CLCDSA [33]	F1
	C-V	Exception Type	ET	K1: Kanade et al. [12]	Acc
	C-C	Code-to-Code Retrieval	CR	P1: POJ-104 [34]	MAP
				C2: CLCDSA [33]	MRR
	NL-C	Code Search	CS	C3: CodeSearchNet (Filtered) [35]	MRR
				A1: AdvTest [35]	MRR
				C4: CoSQA [36], WebQueryTest [35]	MRR
Gen.	C-C	Code Question Answering	QA	F1: FDM [12]	Acc
				C5: CodeTrans [35]	EM/B./C.B.
				T1: TransCoder [37]	CA
				C2: CLCDSA [33]	R.L
				B2: BFP [38]	EM/B./C.B.
	C-C	Bug Fixing	BF	P2: PY150 [39]	EM/ES
				C6: CugLM [40]	EM
				S1: SLM [41]	EM
				S2: Svyatkovskiy et al. [14]	PPL
	C-C	Code Completion	CC	G1: GM [42]	EM/B.
				A3: ATLAS [43]	EM/B.
				C3: CodeSearchNet (Filtered) [35]	B.
				A2: Attn2FC [44]	B.
				D3: DeepCom [45]	B.
	C-NL	Code Summarization	SM	P2: PY150 [39]	EM
				C7: code2seq [46]	EM
				T2: TL-CodeSum [47]	B.
				M1: Miceli-Barone and Sennrich [48]	B.
	NL-C	Code Generation	CG	C8: CONCODE [49]	EM/B./C.B.

参考：PTMの現状の評価

Models	Arch.	Modality	Tasks	PL	Code Understanding Tasks					Code Generation Tasks						
					C-V		C-C	NL-C		CT	BF	C-C	MG	AG	C-NL	NL-C
					DD	CD	ET	CR	CS			QA			SM	CG
RoBERTa [7]	TE	NL Text	MLM	-	D1:61.05	B1:94.9		P1:76.67	C3:61.7 A1:18.33			C4:60.3			C3:16.57	
GPT-2 [9]	TD	NL Text	ULM	-									P2:41.73			17.35
BART [11]	TF	NL Text	DAE	-							11.7					
T5 [10]	TF	NL Text	Seq2Seq MLM	-	D1:61.93						9.7				C3:18.35	18.65
SCeLMo [52]	LSTM	Code	BiLM	Mono	D2:93.12 V1:95.21											
CuBERT [12]	TE	Code	MLM + NSP	Mono	W1:92.46 S3:93.36		79.12		F1:98.09						P2:33.48 C7:52.76 D3:17.41	
GPT-C [14]	TD	Code	ULM	Multi									S2:1.65			
C-BERT [13]	TE	Code	MLM	Mono	D1:57.4											
JavaBERT [15]	TE	Code	MLM	Mono												
CodeGPT-adapted [35]	TD	Code	ULM	Multi									P2:42.37			20.1
DeepDebug [16]	TF	Code	Seq2Seq MLM	Mono							15.05					
CodeBERT [53]	TE	Code + Doc	MLM & RTD	Multi				P1:82.67	C3:69.3	C4:65.7	C5:58.5	10.8			C3:17.83 P2:35.97 C7:56.52 D3:17.87	
GraphCodeBERT [54]	TE	Code + Doc + DFG Nodes	MLM + EP + NA	Multi		B1:97.1		P1:85.16	C3:71.3	C4:68.4	C5:59.1	13.2				
CugLM [40]	TE	Code	IMLM + NSP + ULM	Multi									C6:81.91			
DOBF [55]	TF	Code	MLM & Seq2Seq IMLM	Multi		B1:95.9			A1:38.3		T1:46.35				C3:18.65	
T5-learning [56]	TF	Code & Doc	Seq2Seq MLM	Mono							6.5		28	40.5	A2:15	
PLBART [57]	TF	Code & Post	DAE	Multi	D1:63.18	B1:97.2				C4:65.0	C5:64.8	14.10			C3:18.32	18.75
ProphetNet-Code [58]	TF	Code & Doc	FNP	Multi											C3:18.54	
CoText [59]	TF	Code + Doc	Seq2Seq MLM	Multi	D1:65.99						17.30				C3:18.38 D3:20.49 P2:45.81 C7:67.9	20.1
TreeBERT [28]	TF	Code + AST Paths	TMLM + NOP	Multi												
OSCAR [60]	TE	IR + AEI	MLM + CCL	Mono				P1:49.17								
CodeDisen [61]	LSTM	Code + AST Seq	VGVAE + CLR + PD + ACP	Multi		C2:90.0		C2:43.6			C2:50.08					
CodeT5 [62]	TF	Code + Doc	Seq2Seq MLM / IT / Seq2Seq IMLM / BDG	Multi	D1:65.78	B1:97.2				C4:67.8	C5:66.4	17.79			C3:19.55	22.30
SynCoBERT [63]	TE	Code + Doc + AST Seq	MLM + IT + TEP + MCL	Multi	D1:64.5	B1:97.4		P1:88.24	C3:74.0 A1:38.1		C5:60.85					
SPT-Code [29]	TF	Code + Names + AST Seq	CAP & MASS & MNG	Multi					C3:71.5		C5:62.18	14.2	S1:19.09		C3:15.0 T2:49.1 M1:36.1	
UniXcoder [64]	TE	AST Seq + Doc	MLM / ULM / Seq2Seq MLM / MCL / CMG	Multi		B1:95.2		P1:90.52	C3:74.4 A1:41.3						C3:19.30	22.60

→全体として十分に比較されているとは言えない

評価結果 A

コード理解タスク：欠陥検出

Model	DD	
	Acc	
	Cur	New
Transformer		64.40
RoBERTa	61.05	64.47
GPT-2	-	63.22
BART	-	63.81
T5	61.93	61.87
CuBERT	-	64.25
GPT-C	-	63.77
C-BERT	57.4	64.05
JavaBERT	-	64.50
CodeGPT-adapted	-	65.64
DeepDebug	-	64.18
CodeBERT	-	65.02
GraphCodeBERT	-	<u>65.92</u>
CugLM	-	64.19
DOBF	-	63.86
T5-learning	-	63.60
PLBART	63.18	64.21
ProphetNet-Code	-	63.57
CoText	65.99	65.68
TreeBERT	-	65.76
CodeT5	65.78	65.82
SynCoBERT	64.5	66.25
SPT-Code	-	64.88
UniXcoder	-	65.64

- SynCoBERTはCoTextを上回り新たなSOTAとなった
(元はCoTextがSOTA) ※SOTA→最先端, 最高水準の意
- アーキテクチャ：
上位2モデルはどちらもTE (Transformer-encoder) ベース.
TEがこのタスクに適したArch.であると結論づけるには証拠不十分
- モーダリティ：コード構造とNLの双方が性能に+の影響
得られる情報の使用方法も重要
例) TFベースのTreeBERTはAST木を使用することでTFのベースモデルを凌駕

評価結果 B

コード理解タスク：クローン検出

Model	CD	
	F1	
	Cur	New
Transformer		89.27
RoBERTa	94.9	95.35
GPT-2	-	96.22
BART	-	95.11
T5	-	94.86
CuBERT	-	94.78
GPT-C	-	95.46
C-BERT	-	95.00
JavaBERT	-	96.57
CodeGPT-adapted	-	96.65
DeepDebug	-	95.90
CodeBERT	-	96.77
GraphCodeBERT	97.1	97.11
CugLM	-	96.44
DOBF	95.9	96.84
T5-learning	-	96.38
PLBART	97.2	97.01
ProphetNet-Code	-	96.05
CoText	-	95.96
TreeBERT	-	96.51
CodeT5	97.2	97.18
SynCoBERT	97.4	97.55
SPT-Code	-	96.40
UniXcoder	95.2	96.32

- PLBARTの順位が下がったこと以外現状の結果と同じ
- 事前学習タスクが単一であるより複数使用されている方がこのタスクにおいて優れていると推測（上位2位のSynCoBERTとCodeT5が4つに対し，PLBARTは単一）
- 新しい結果によってTD（Transformer Decoder）ベースの性能が比較できるように
以前最高のTDベースPTMとされたCodeGPT-adaptedは7位であった。

評価結果 C

コード理解タスク：例外タイプ

Model	ET	
	Acc	
	Cur	New
Transformer		48.98
RoBERTa	-	76.94
GPT-2	-	75.54
BART	-	73.68
T5	-	74.75
CuBERT	79.12	79.90
GPT-C	-	78.26
C-BERT	-	74.57
JavaBERT	-	67.66
CodeGPT-adapted	-	76.71
DeepDebug	-	73.50
CodeBERT	-	81.25
GraphCodeBERT	-	83.26
CugLM	-	79.01
DOBF	-	79.04
T5-learning	-	69.85
PLBART	-	77.93
ProphetNet-Code	-	79.37
CoText	-	77.21
TreeBERT	-	78.08
CodeT5	-	85.00
SynCoBERT	-	82.70
SPT-Code	-	77.11
UniXcoder	-	<u>83.47</u>

- 13のSE評価タスクの中で唯一のマルチラベル分類タスク
- 注目すべきはTEベースではSOTA性能が達成されず，TF（Transformer，テキストからテキストへ変換）のCodeT5が最良であったこと
- 他のArch.で最良とされるモデルは以下の通り
TEベース UniXcoder：2位
TDベース GPT-C：10位
- CodeT5は各ラベルを一意の文字列にマッピングする．例外のインデックスではなく，テキストを予測する
対してTEベースでは，事前学習タスクの大部分がバイナリ分類であるため，不向きである可能性がある

評価結果 D

コード理解タスク：Code to Code検索

Model	CR	
	MAP	
	Cur	New
Transformer		64.27
RoBERTa	76.67	80.20
GPT-2	-	53.30
BART	-	79.63
T5	-	69.16
CuBERT	-	76.87
GPT-C	-	55.23
C-BERT	-	72.91
JavaBERT	-	77.44
CodeGPT-adapted	-	72.63
DeepDebug	-	73.51
CodeBERT	82.67	85.61
GraphCodeBERT	85.16	87.73
CugLM	-	83.32
DOBF	-	87.31
T5-learning	-	80.82
PLBART	-	85.02
ProphetNet-Code	-	79.82
CoText	-	86.65
TreeBERT	-	85.54
CodeT5	-	87.53
SynCoBERT	88.24	88.52
SPT-Code	-	86.54
UniXcoder	90.52	90.55

- 新しい結果よりTEベースが他のArch.より有利である

- 最も性能の良い他のモデル

TFベース CodeT5：4位

TDベース CodeGPT-adapted：20位

評価結果 E

コード理解タスク：コード検索

Model	CS	
	MRR	
	Cur	New
Transformer		3.12
RoBERTa	18.33	18.82
GPT-2	-	16.38
BART	-	16.65
T5	-	16.97
CuBERT	-	22.26
GPT-C	-	24.39
C-BERT	-	25.34
JavaBERT	-	25.02
CodeGPT-adapted	-	25.97
DeepDebug	-	30.58
CodeBERT	-	38.21
GraphCodeBERT	-	38.76
CugLM	-	36.20
DOBF	38.3	38.56
T5-learning	-	37.98
PLBART	-	38.70
ProphetNet-Code	-	37.64
CoText	-	38.13
TreeBERT	-	39.60
CodeT5	-	40.03
SynCoBERT	38.1	39.99
SPT-Code	-	37.05
UniXcoder	41.3	41.57

- 現状と変わらず，このタスクではTEベースのUniXcoderが最良であった。
- TEベースとTFベースのモデルが交互にランクイン
→このタスクでは同等の性能を発揮

評価結果 F

コード理解タスク：コード質問応答

Model	QA	
	MRR	
	Cur	New
Transformer		52.89
RoBERTa	1 60.3	60.28
GPT-2	-	58.06
BART	-	55.57
T5	-	45.63
CuBERT	-	54.33
GPT-C	-	50.32
C-BERT	-	54.81
JavaBERT	-	54.04
CodeGPT-adapted	-	54.24
DeepDebug	-	57.39
CodeBERT	65.7	65.90
GraphCodeBERT	68.4	68.55
CugLM	-	61.44
DOBF	-	61.31
T5-learning	-	60.21
PLBART	65.0	65.01
ProphetNet-Code	-	63.73
CoText	-	68.70
TreeBERT	-	64.98
CodeT5	67.8	67.91
SynCoBERT	-	69.19
SPT-Code	-	64.55
UniXcoder	70.1	70.30

- 現状と変わらず，このタスクではTEベースのUniXcoderが最良であった。
- UniXcoderの事前学習タスクであるMCLは，2つの入力が互いに一致するかどうかを区別することを目的としており，このタスクの目標と近いためスコアが高かったと考えられる。

評価結果 G：コード翻訳

Model	Code Translation											
	Java->C#						C#->Java					
	EM		BLEU		CodeBLEU		EM		BLEU		CodeBLEU	
	Cur	New	Cur	New	Cur	New	Cur	New	Cur	New	Cur	New
Transformer	33.0	40.8	55.84	60.22	63.74	67.10	37.9	43.9	50.47	54.86	61.59	61.84
RoBERTa	-	58.9	-	79.70	-	83.77	-	59.5	-	73.14	-	78.63
GPT-2	-	54.4	-	65.39	-	76.92	-	56.0	-	69.41	-	71.24
BART	-	49.5	-	67.91	-	74.25	-	55.2	-	72.32	-	70.80
T5	-	45.3	-	69.23	-	76.05	-	53.2	-	73.84	-	71.52
CuBERT	-	55.6	-	75.15	-	79.30	-	55.6	-	70.07	-	75.31
GPT-C	-	60.9	-	77.91	-	82.48	-	59.6	-	72.94	-	78.18
C-BERT	-	55.4	-	74.65	-	81.63	-	56.4	-	70.28	-	75.73
JavaBERT	-	61.1	-	80.74	-	80.45	-	58.3	-	70.10	-	77.14
CodeGPT-adapted	-	62.0	-	80.21	-	85.47	-	60.3	-	72.93	-	79.02
DeepDebug	-	59.5	-	81.46	-	83.95	-	63.8	-	75.10	-	82.43
CodeBERT	59.0	61.2	79.92	81.16	85.10	85.29	58.0	60.1	72.14	73.73	79.41	80.11
GraphCodeBERT	59.4	62.6	80.58	81.24	-	85.34	58.8	61.5	72.64	73.67	-	80.63
CugLM	-	60.8	-	78.34	-	83.65	-	61.6	-	73.95	-	78.06
DOBF	-	64.8	-	80.27	-	82.77	-	64.6	-	75.44	-	80.53
T5-learning	-	62.9	-	78.19	-	81.13	-	64.8	-	75.64	-	81.09
PLBART	64.6	67.8	83.02	84.75	87.92	88.16	65.0	67.4	78.35	79.75	85.27	85.05
ProphetNet-Code	-	62.5	-	80.38	-	81.64	-	64.5	-	75.68	-	81.04
CoText	-	65.7	-	83.35	-	85.63	-	65.4	-	77.98	-	82.31
TreeBERT	-	62.1	-	81.72	-	84.34	-	64.2	-	76.33	-	81.17
CodeT5	65.9	67.2	84.03	84.97	-	87.50	66.9	66.3	79.87	79.67	-	83.70
SynCoBERT	60.4	64.1	80.75	82.52	84.85	85.60	61.3	63.8	76.52	77.53	82.22	82.36
SPT-Code	64.1	66.6	90.34	83.24	-	85.15	60.3	63.9	86.10	78.82	-	85.33
UniXcoder	-	64.5	-	81.66	-	85.60	-	64.1	-	77.37	-	82.56

- "Java to C#" と "C# to Java" の2パターン
- PLBARTとCodeT5がそれぞれTop2
- Top5はすべてTFモデルであった。
TEモデルは順位を下げた
- モーダリティにNLが入っていることが重要である。

評価結果 H：バグ修正

Model	Bug Fixing											
	small						medium					
	EM		BLEU		CodeBLEU		EM		BLEU		CodeBLEU	
	Cur	New	Cur	New	Cur	New	Cur	New	Cur	New	Cur	New
Transformer	14.7	14.63	77.21	76.92	-	73.88	3.7	8.46	89.25	89.23	-	86.86
RoBERTa	-	13.88	-	79.72	-	78.31	-	9.09	-	88.69	-	84.05
GPT-2	-	14.97	-	64.42	-	68.10	-	5.05	-	74.11	-	72.42
BART	16.7	15.60	-	71.34	-	72.43	6.7	6.97	-	82.37	-	81.85
T5	15.3	14.34	-	69.71	-	73.10	4.11	6.49	-	78.22	-	78.20
CuBERT	-	14.87	-	74.93	-	75.28	-	8.92	-	86.12	-	83.09
GPT-C	-	13.08	-	70.06	-	71.83	-	8.26	-	85.41	-	82.47
C-BERT	-	14.04	-	73.19	-	74.54	-	9.37	-	85.57	-	83.87
JavaBERT	-	15.39	-	78.98	-	76.02	-	9.41	-	85.33	-	84.42
CodeGPT-adapted	-	13.66	-	76.07	-	77.13	-	11.00	-	85.28	-	84.55
DeepDebug	18.7	18.13	-	76.64	-	76.91	11.4	11.09	-	87.10	-	85.80
CodeBERT	16.4	14.66	77.42	78.41	-	78.09	5.2	9.72	90.07	86.94	-	83.88
GraphCodeBERT	17.3	16.85	80.02	79.61	-	79.68	9.1	10.14	91.31	87.63	-	85.33
CugLM	-	13.78	-	75.36	-	75.20	-	9.08	-	85.20	-	84.82
DOBF	-	15.45	-	75.52	-	74.62	-	10.28	-	87.93	-	85.01
T5-learning	10	17.62	-	77.05	-	76.30	3	10.94	-	88.46	-	86.42
PLBART	19.21	19.40	77.02	78.03	-	77.58	8.98	11.05	88.5	88.48	-	86.67
ProphetNet-Code	-	17.23	-	75.40	-	75.60	-	10.75	-	86.82	-	84.19
CoTexT	21.58	21.33	77.28	77.20	77.38	77.75	13.03	13.37	88.68	87.13	84.41	85.14
TreeBERT	-	20.73	-	79.38	-	75.17	-	12.89	-	89.05	-	87.15
CodeT5	21.61	21.65	77.43	77.55	-	77.24	13.96	14.30	87.64	89.23	-	87.05
SynCoBERT	-	20.32	-	78.81	-	78.56	-	11.17	-	87.94	-	86.10
SPT-Code	17.54	18.59	75.10	78.51	-	74.97	10.86	12.06	87.88	88.37	-	84.35

- Top4はCodeT5, CoTexT, TreeBERT, UniXcoder.
- Top3のArch.はTFベースであった
TEとTDベースの順位は低い
- Masked Language Modelの事前学習
タスクを使っているモデルは有利。
上位ほとんどのモデルが使用。

評価結果Ⅰ：コード補完

Model	Code Completion			
	EM		ES	
	Cur	New	Cur	New
Transformer	-	37.71	-	67.95
RoBERTa	-	<u>39.01</u>	-	<u>68.98</u>
GPT-2	41.73	41.77	-	70.30
BART	-	30.67	-	56.17
T5	-	28.58	-	55.24
CuBERT	-	38.32	-	66.97
GPT-C	-	<u>42.82</u>	-	<u>71.35</u>
C-BERT	-	41.07	-	67.82
JavaBERT	-	39.16	-	67.67
CodeGPT-adapted	42.37	43.80	-	72.54
DeepDebug	-	39.68	-	65.28
CodeBERT	-	40.77	-	68.58
GraphCodeBERT	-	40.26	-	69.88
CugLM	-	42.94	-	71.89
DOBF	-	39.35	-	69.30
T5-learning	-	38.06	-	66.58
PLBART	-	<u>41.74</u>	-	68.42
ProphetNet-Code	-	39.66	-	67.24
CoText	-	40.36	-	70.16
TreeBERT	-	41.73	-	70.14
CodeT5	-	40.52	-	<u>71.29</u>
SynCoBERT	-	41.52	-	70.26
SPT-Code	-	40.20	-	68.97
UniXcoder	-	41.69	-	69.84

- CodeGPT-adaptedがSOTAとなった
TDモデルによりSOTAが達成されたのは今回の比較で唯一.
- Unidirectionalなモデル（ CodeGPT-adapted , CugLM, GPT-C ）がTop3で有利
- モーダリティのうち, NLとコード構造が優位な役割を果たさない唯一のタスク

評価結果 J：Assert Generation

Assertion：コード内で特定の条件が真であることを確認するための文。
条件が満たされない場合エラーを引き起こし，例外の場所等を迅速に特定できる

- Top5はすべてTFベースであった。
上からPLBART, TreeBERT, SPT-Code, T5-learning, CodeT5
- TEとTDは比較的下位であった。
- TreeBERT以外のTop5は全てモーダリティにNLが入っており，有意な役割を果たしている。

評価結果 K：Mutant Generation

- Top3はCodeT5, TreeBERT, PLBART.

これらはコードに加えて、NLまたはコード構造のいずれかを入力の1つとして使用しているため、モーダリティのうちNLとコード構造が+の影響

- DAE (PLBART), TMLM, NOP (TreeBERT), CAP (SPT-Code) といった構造を意識した事前学習タスクが+の影響.

評価結果 L：コード要約

Model	Code Summarization											
	Java		Py		JS		PHP		Go		Ruby	
	BLEU		BLEU		BLEU		BLEU		BLEU		BLEU	
	Cur	New	Cur	New	Cur	New	Cur	New	Cur	New	Cur	New
Transformer	16.26	16.37	15.81	16.47	11.59	10.26	22.12	23.41	16.38	16.46	11.18	11.09
RoBERTa	16.47	17.38	18.14	17.49	11.90	11.75	24.02	24.42	17.72	17.63	11.17	11.26
GPT-2	-	18.62	-	18.92	-	14.13	-	23.91	-	17.59	-	12.03
BART	-	18.98	-	19.37	-	14.68	-	24.46	-	18.80	-	13.65
T5	18.35	18.87	19.26	19.57	14.57	14.45	24.59	24.32	19.17	18.70	14.18	13.72
CuBERT	-	16.75	-	17.77	-	11.34	-	22.76	-	16.09	-	10.46
GPT-C	-	17.18	-	17.78	-	12.01	-	23.42	-	16.96	-	10.54
C-BERT	-	17.44	-	18.39	-	13.14	-	23.90	-	17.47	-	12.14
JavaBERT	-	18.23	-	17.57	-	11.91	-	22.87	-	17.13	-	10.94
CodeGPT-adapted	-	17.68	-	18.46	-	12.91	-	24.68	-	17.38	-	12.39
DeepDebug	-	19.00	-	18.85	-	14.39	-	23.37	-	17.68	-	13.27
CodeBERT	17.65	18.61	19.06	19.23	14.90	14.75	25.16	24.70	18.07	18.26	12.16	12.53
GraphCodeBERT	-	18.93	-	19.39	-	14.90	-	25.64	-	18.50	-	12.63
CugLM	-	18.04	-	18.20	-	14.07	-	24.66	-	18.53	-	11.47
DOBF	19.05	19.18	18.24	18.41	-	13.22	-	23.83	-	18.28	-	13.21
T5-learning	-	18.84	-	18.23	-	13.18	-	23.10	-	17.29	-	12.51
PLBART	18.45	19.31	19.30	19.41	15.56	15.73	23.58	24.47	18.91	19.01	14.11	14.15
ProphetNet-Code	19.39	19.29	17.87	18.20	16.60	15.95	24.57	24.28	18.43	18.31	14.37	14.39
CoTexT	19.10	19.19	19.52	19.72	14.77	15.08	24.47	24.57	19.37	19.13	13.07	14.28
TreeBERT	-	18.90	-	19.44	-	15.05	-	23.82	-	18.74	-	13.57
CodeT5	20.31	20.35	20.01	20.17	16.16	16.75	26.03	25.97	19.56	19.68	15.24	15.36
SynCoBERT	-	18.89	-	18.74	-	14.57	-	25.55	-	18.36	-	13.91
SPT-Code	-	18.90	-	19.71	-	15.28	-	24.78	-	19.17	-	14.38
UniXcoder	-	19.42	-	18.64	-	14.27	-	25.70	-	18.59	-	14.32

- Top4はCodeT5, ProphetNet-Code, CoTexT, SPT-Codeで, これらはTFベース. このタスクではTFが有利
- 最高のTEとTDベースのものは5位 (UniXcoder) と15位 (GPT-2) で以前の結果とは一致せず.
- モーダリティNLのものが有利であった.
これは, 事前学習でNLを生成していることと, ソースコードの「自然さ」が関係していると考えられる
- クロスモーダル (コードとNL) を意識した事前学習が行われているものが有利であった

評価結果 M：コード生成

Model	Code Generation					
	EM		BLEU		CodeBLEU	
	Cur	New	Cur	New	Cur	New
Transformer	-	6.10	-	21.67	-	26.98
RoBERTa	-	19.30	-	<u>31.92</u>	-	35.40
GPT-2	17.35	17.55	25.37	23.62	29.69	29.93
BART	-	<u>18.90</u>	-	31.38	-	<u>33.72</u>
T5	18.65	18.70	32.74	32.02	35.95	33.26
CuBERT	-	19.05	-	30.14	-	32.82
GPT-C	-	<u>19.85</u>	-	30.45	-	33.10
C-BERT	-	19.80	-	33.62	-	35.99
JavaBERT	-	18.45	-	34.62	-	36.93
CodeGPT-adapted	20.10	20.15	32.79	<u>35.94</u>	35.98	<u>37.27</u>
DeepDebug	-	18.40	-	36.52	-	38.90
CodeBERT	-	21.15	-	31.45	-	35.26
GraphCodeBERT	-	21.00	-	34.33	-	37.55
CugLM	-	21.80	-	33.70	-	35.91
DOBF	-	20.35	-	35.26	-	37.41
T5-learning	-	18.95	-	35.76	-	38.30
PLBART	18.75	19.85	36.69	36.63	38.52	39.29
ProphetNet-Code	-	21.70	-	37.67	-	39.79
CoTexT	20.10	21.80	36.51	38.74	39.49	40.63
TreeBERT	-	22.05	-	38.27	-	39.73
CodeT5	22.30	23.40	40.73	40.75	43.20	43.40
SynCoBERT	-	21.35	-	38.39	-	<u>41.21</u>
SPT-Code	-	20.00	-	37.91	-	39.90
UniXcoder	22.60	<u>22.65</u>	-	38.73	-	40.86

- SOTAはTEベース（UniXcoder）からTFベース（CodeT5）に変化。
最良のTEベース（UniXcoder）とTDベース（CodeGPT-adapted）モデルはそれぞれ2位と11位。
- CodeT5はコード要約でもSOTAであるものの、コード理解とコード生成は根本的に性質が異なるため、このタスクでのT5とBART(NLのみで事前学習済み)はコード要約の順位よりも低い。
- Top3のうちCodeT5のみがコードとNLの両方を使用。UniXcoderはコード構造とNL、TreeBERTはコード構造とコードを使用する。
→コードとNLのモダリティの重要性はコードようやくほど顕著でない
- コード構造に関する事前学習が+の影響